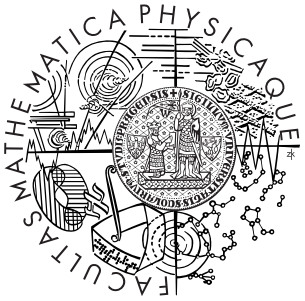
Univerzita Karlova v Praze
Matematicko-fyzikální fakulta

# BAKALÁŘSKÁ PRÁCE




Jakub Repický

## Systémy automatického dokazování

Katedra algebry

Vedoucí bakalářské práce: Mgr. David Stanovský, PhD.
Studijní program: Informatika, Obecná informatika

2009

Ďakujem doktorovi Stanovskému za trpezlivosť a cenné rady pri písaní tejto práce. Ďakujem svojim rodičom za to, že mi umožnili študovať.

Prohlašuji, že jsem svou bakalářskou práci napsal samostatně a výhradně s použitím citovaných pramenů. Souhlasím se zapůjčováním práce a jejím zveřejňováním.

V Praze dne 27. 5. 2009 Jakub Repický

# Obsah

Název práce: Systémy automatického dokazování
Autor: Jakub Repický
Katedra (ústav): Katedra algebry
Vedoucí bakalářské práce: Mgr. David Stanovský, PhD.
e-mail vedoucího: stanovsk@karlin.mff.cuni.cz

Abstrakt: V grupe $G$ definujeme operáciu konjugácie rovnicou $x * y = xyx^{-1}$ pre všetky $x, y \in G$. Študujeme rovnice platné vo všetkých grupoidoch $G(*)$, kde $G$ je grupa. K doteraz známym výsledkom patria rovnice idempotencie, ľavej distributivity a množina rovníc, ktorú objavil D. Larue. Pomocou skriptov v jazyku Perl generujeme rovnice konjugácie platné vo všetkých $G(*)$ a pomocou systémov automatického dokazovania viet a hľadania modelov (Prover9, Waldmeister, Mace4) medzi nimi hľadáme rovnice nezávislé na doterajších výsledkoch alebo také, ktoré by zjednodušovali niektoré z Larueových rovníc.

Klíčová slova: automatické dokazování vět, konjugace, volná grupa

Title: Systems of Automated Theorem Proving
Author: Jakub Repický
Department: Department of Algebra
Supervisor: Mgr. David Stanovský, PhD.
Supervisor's e-mail address: stanovsk@karlin.mff.cuni.cz

Abstract: Let $G$ be a group. We define an operation of conjugation by $x * y = xyx^{-1}$ for all $x, y \in G$. We study equations that hold in all grupoids $G(*)$, $G$ a group. These are some of such equations: left distributivity, idempotency and a set of equations found by D. Larue. We generate equations of conjugation true in all $G(*)$ by scripts written in Perl and we search for equations either not following from equations known so far or simplifying some of the Larue's equations by systems of automated theorem proving and model building.

Keywords: automated theorem proving, conjugation, free group

# Kapitola 1

# Rovnicová teória konjugácie

## 1.1 Slovo, voľná grupa

Nech $G$ je grupa. **Slovom** nad množinou $X \subseteq G$ rozumieme zápis produktu prvkov z $X$ a prvkov k nim inverzným v tvare $x_1^{\epsilon_1} x_2^{\epsilon_2} \dots x_n^{\epsilon_n}$, kde $x_1, x_2, \dots x_n \in X$ a $\epsilon_i \in \{1, -1\}$ pre $i = 1, 2, \dots, n$.

Slovo $w$ nazývame **redukované**, ak neobsahuje podslovo $xx^{-1}$.

Voľnú grupu $G(\cdot, {}^{-1}, e)$ nad abecedou $X$ definujeme takto:

- $G$ je množina všetkých slov nad množinou $X$
- operáciu $w_1 \cdot w_2$ definujeme ako redukciu zreťazenia slov $w_1$ a $w_2$ (redukciou slova $w$ myslíme opakované odstraňovanie podslov $xx^{-1}$ zo slova $w$, kým sa tam také podslovo vyskytuje)
- operáciu inverzného prvku definujeme rovnicou

$$(x_1^{\epsilon_1} \dots x_n^{\epsilon_n})^{-1} = x_n^{-\epsilon_n} \dots x_1^{-\epsilon_1}$$

- neutrálnym prvkom $e$ je prázdne slovo

## 1.2 Konjugácia

Pojem konjugácie zohráva dôležitú úlohu v teórii grúp, používa sa pri štúdiu permutačných grúp, neabelovských grúp alebo automorfizmov. Na konjugáciu môžeme nahliadať ako na reláciu, unárne, alebo binárne zobrazenie.

Ak $G$ je grupa, o prvkoch $x, y \in G$ hovoríme, že sú konjugované, ak existuje prvok $g \in G$ taký, že platí $gxg^{-1} = y$.

Je zrejmé, že relácia konjugácie je ekvivalencia. Množiny $\{gag^{-1} : g \in G\}$ pre $a \in G$ sa nazývajú triedami konjugácie.

Zobrazenie $\varphi_y : G \mapsto G$ pre pevné $y \in G$, definované $\varphi_y : x \mapsto yxy^{-1}$ sa nazýva vnútorný automorfizmus.

V tejto práci sa však budeme zaoberať binárnou operáciou konjugácie, ktorú definujeme predpisom

$$x * y = xyx^{-1}. \tag{1.1}$$

Budeme používať nasledujúce značenie: keďže binárna operácia grupy je z definície asociatívna, pri zápise slov vo voľnej grupe nepoužívame symbol zreťazenia ani zátvorky. Termy konjugácie štandardne zapisujeme v tvare $t_1 * t_2$, napr. $x * (y * z)$. Ak nedôjde k zmäteniu, riadime sa konvenciou, že symboly v termoch konjugácie sú asociované zprava a teda termy zapisujeme skratkou bez operátora a zátvoriek, napr. $xyz = x(yz) = x * (y * z)$.

Ak $G$ je grupa a $*$ operácia konjugácie, podľa štandardnej definície pravdivosti hovoríme, že rovnica $R \equiv t_1(x_1, \ldots, x_n) = t_2(x_1, \ldots, x_n)$ platí v grupoide $G(*)$ (resp., že grupoid $G$ spĺňa rovnicu $R$), ak pre všetky ohodnotenia $e$ premenných $x_1, \ldots, x_n$ platí $t_1[e] = t_2[e]$.

Dôsledkom Tvrdenia 3 A. Drápala, T. Kepku a M. Musílka [1] je, že rovnica $t_1(x_1, \ldots, x_n) = t_2(x_1, \ldots, x_n)$, platí vo všetkých grupoidoch $G(*)$, $G$ grupa, práve vtedy, keď $t_1(x_1, \ldots, x_n) = t_2(x_1, \ldots, x_n)$ platí ako slovo vo voľnej grupe.

Z tohto plynie nasledujúce pozorovanie:

**Pozorovanie.** *Vo voľnej grupe a teda v $G(*)$ pre všetky grupy $G$ platia tieto rovnice: $I \equiv x * x = x$ (idempotencia) a $LD \equiv x * (y * z) = (x * y) * (x * z)$ (ľavá distributivita).*

*Dôkaz.*

Idempotencia:

$$x * x \approx xxx^{-1} = x \approx x$$

Ľavá distributivita:

$$x * (y * z) \approx x * (yzy^{-1}) \approx xyzy^{-1}x^{-1}$$

$$(x * y) * (x * z) \approx (xyx^{-1}) * (xzx^{-1}) \approx xyx^{-1}xzx^{-1}xy^{-1}x^{-1} = xyzy^{-1}x^{-1}$$

Prepísali sme ľavú a pravú stranu rovníc na slová vo voľnej grupe podľa definície 1.1 a vidíme, že sa rovnajú. □

Navyše existuje nekonečná množina nezávislých rovníc, ktoré neplynú z idempotencie a ľavej distributivity a ktorú objavil D. Larue [2]. Nazveme ich *Larueove rovnice*:

$$\begin{array}{lcrcl} Larue_1 & \equiv & (xy * y)(xz) & = & (xy)(yx * z) \\ Larue_2 & \equiv & (yxy * xy * y)(yx * z) & = & (yxy * xy)(yyx * z) \\ Larue_3 & \equiv & (yyxy * yxy * xy * y)(yyx * z) & = & (yyxy * yxy * xy)(yyyx * z) \end{array}$$

atď.

## 1.3 Ciele práce

Základným otvoreným problémom operácie konjugácie (formulovaným Tomášom Kepkou v 80-tych rokoch) je zistiť, ktoré rovnice platia vo všetkých grupoidoch $G(*)$, špeciálne, či existuje konečná báza tejto rovnicovej teórie. Čiastočné výsledky boli dosiahnuté v článkoch [1][2][3], napriek tomu sa v súčasnosti zdá, že riešenie je v nedohľadne. Cieľom tejto práce je pomôcť výskumu úplným prehľadaním menej komplikovaných rovníc. Hlavné ciele tejto práce môžeme formulovať takto:

- Nájsť nové rovnice $R$, ktoré platia vo voľnej grupe a ktoré sú nezávislé na $I$, $LD$ a Larueových rovniciach, t.j. pre všetky kladné $n$ existuje grupoid $G(*)$, ktorý spĺňa konjunkciu $LD \,\&\, I \,\&\, Larue_1 \,\&\, \ldots \,\&\, Larue_n$ a nespĺňa rovnicu $R$.

- Zjednodušiť Larueove rovnice, t.j. nájsť takú rovnicu $R$, pre ktorú by pre nejaké $n > 1$ platilo

  $$LD \,\&\, I \,\&\, R \Rightarrow Larue_1 \,\&\, \ldots \,\&\, Larue_n,$$

  alebo aspoň pre nejaké $i, j > 0, i \neq j$:

  $$LD \,\&\, I \,\&\, R \,\&\, Larue_i \Rightarrow Larue_j.$$

Automatické dokazovače viet samozrejme pracujú s konečnými množinami rovníc a pri komplikovaných rovniciach sú neefektívne, používame ich na hľadanie kandidátov pre ďalšie matematické skúmanie. Preto pri výpočtoch testovanie obmedzujeme na prvé dve Larueove rovnice.

Konkrétne teda pre vygenerované rovnice $R$ platné vo voľnej grupe nástrojmi automatického dokazovania testujeme nasledujúce:

1. $\neg(LD \,\&\, I \,\&\, Larue_1 \,\&\, Larue_2 \Rightarrow R)$
2. $LD \,\&\, I \,\&\, R \Rightarrow Larue_1 \,\&\, Larue_2$
3. $LD \,\&\, I \,\&\, R \,\&\, Larue_1 \Rightarrow Larue_2$
4. $LD \,\&\, I \,\&\, R \,\&\, Larue_2 \Rightarrow Larue_1$

# Kapitola 2

# Užívateľská dokumentácia skriptov

## 2.1 Systémové požiadavky a závislosti

- Unixový operačný systém (program testovaný na Linuxe, ale mal by fungovať aj na ostatných systémoch orientovaných na normu *POSIX*)
- Perl
- Automatický dokazovač viet *Prover9*
- Hľadač modelov *Mace4*
- Automatický dokazovač rovníc na báze prepisovacích systémov *Waldmeister*

## 2.2 Generátor rovníc

Skript `eq_generator.pl` slúži na generovanie binárnych termov a rovníc konjugácie platných v grupách do zadanej hĺbky. Ponúka dve akcie:

`-gen-terms` generuje alebo rozširuje databázu termov konjugácie a súčasne obnovuje index termov: kľúčom je slovo vo voľnej grupe a hodnotou je zoznam už vygenerovaných konjugačných termov, ktoré sa mu po prevedení na slovo a redukcii rovnajú

`-extract-equations` z databázy slov vygeneruje zoznam rovníc, ktoré sú splnené vo voľnej grupe

**Príklad 1.**

```
$ ./eq_generator -gen-terms -f terms.db -w words.db -depth 5
```

Vygeneruje (alebo rozšíri, v prípade, že databáza `terms.db` existuje) zoznam termov konjugácie do súboru `terms.db` a ich index podľa slov do súboru `words.db`.

Parameter `-depth` určuje hĺbku generovania, resp. zložitosť termov (množina hĺbky 1 je vstupná množina termov, množina hĺbky $n$ rozšíri množinu hĺbky $n-1$ o všetky dvojprvkové kombinácie jej prvkov, ktoré spĺňajú určité podmienky (kapitola 3.1)).

**Príklad 2.**

```
$ ./eq_generator -extract-eq -i words.db -o equations.db
```

Vygeneruje zoznam rovníc z indexu termov z predchádzajúceho príkladu.

## 2.3 Dokazovač rovníc

Skript `eq_prover.pl` funguje ako filter súborov s rovnicami. Funguje v dvoch módoch: dokazovač a hľadač modelov. Skript prechádza rovnice zo vstupného súboru a pre každú z nich spustí hľadanie definované kombináciou volieb:

`-prover` zapne mód dokazovača

`-model-finder` zapne mód hľadania modelov

`-input` vstupný súbor s rovnicami

`-output` výstupný súbor

`-axioms` definuje súbor s množinou axiómov

`-goal` definuje súbor s cieľovou rovnicou. Ak je tento prepínač zapnutý, rovnice zo vstupného súboru sa vo vstupných súboroch dokazovačov pridajú k množine axiómov. Inak sa použijú ako cieľová rovnica, ktorú treba dokázať, resp. vyvrátiť

`-successful-results` ak zapnuté, tak rovnice, pre ktoré hľadanie dôkazu, alebo modelu skončí úspechom, sa pridajú do výstupného súboru. V opačnom prípade sa do výstupného súboru dostanú iba rovnice, pre ktoré hľadanie skončí neúspechom

`-timeout` definuje časový limit pre hľadanie pomocou externých dokazovačov

Okrem výstupného súboru skript generuje záznamy práce dokazovačov, ktoré ukladá do adresára `data/ATP_logs/TIMESTAMP/`, kde `TIMESTAMP` je čas spustenia skriptu v tvare `YYYY-MM-DD_HH:MM:SS`.

Pracujeme s rovnicami v postfixovom zápise. Ako binárny operátor používame znak *. Na každom riadku súboru, okrem posledného musí byť rovnica. Posledný riadok súboru je rezervovaný pre číslo označujúce počet spracovaných rovníc. Tak je možné proces prerušiť a pokračovať v hľadaní neskôr. Pre súbor s axiómami, alebo cieľom používame iný formát: na každom riadku musí byť rovnica.

Niektoré množiny axiómov sa nachádzajú v priečinku `data` v prílohe (viď kapitolu 5).

**Príklad 3.**

```
./eq_prover.pl -prover -i equations.txt -o ld+i-noproof.txt\
-t 10 -axioms data/ld+i.txt
```

Zo súboru `equations.txt` vyfiltruje tie rovnice, pre ktoré externé dokazovače nenájdu dôkaz z idempotencie a ľavej distributivity v časovom limite 10 sekúnd.

## 2.4 Nástroje

Projekt obsahuje aj niekoľko pomocných skriptov:

`./eq_postfix_to_infix.pl file1 [ file2 [... fileN]]`
prevedie rovnice zo vstupných súborov do infixového zápisu a vypíše ich na štandardný výstup

`./eq_infix_to_postfix.pl file1 [ file2 [... fileN]]`
podobný skript pre prevod z infixového do postfixového zápisu (teda do vstupného formátu pre skript `eq_prover.pl`)

`./gen_larue.pl n`
: vygeneruje $n$-tú Larueovu rovnicu

`./list_substract.pl list1 list2 difference`
: urobí množinový rozdiel súborov s rovnicami `list1` a `list2` a výsledok zapíše do súboru `difference`

## 2.5 Konfigurácia

Konfigurácia je uložená v module `src::Config` (súbor `src/Config.pm`). Obsahuje nastavenia dokazovačov, východzie názvy vstupných a výstupných súborov, pracovných adresárov, množinu atomických termov. Niektoré dôležité premenné:

`DEPTH` východzia hodnota hĺbky generovania

`ATP_LOGS_DIR` názov adresára, do ktorého sa ukladajú vstupné a výstupné súbory dokazovačov

`[P9|WM|M4]_[IN|OUT]_DIR` názvy pracovných adresárov dokazovačov

`[P9|WM|M4]_EXECUTABLE` cesty k spustiteľným súborom dokazovačov

`[P9|WM|M4]_TIMEOUT` východzia hodnota časového limitu pre dokazovače

`EQ_PREFIX` prefix názvov vstupných a výstupných súborov dokazovačov

# Kapitola 3

# Programátorská dokumentácia

## 3.1 Generovanie termov konjugácie a slov

Aby sme generovali iba také rovnice, ktoré platia v grupe, udržiavame spolu s databázou konjugačných termov aj databázu redukovaných slov (používame pojmy z kapitoly 1). Každý vygenerovaný term konjugácie zaindexujeme príslušným slovom (využívame perlovské asociatívne pole). Termy, ktoré sa rovnajú rovnakému slovu oddeľujeme znakom ‘,’ a reťazíme.

Term je v programe reprezentovaný v postfixe ako reťazec znakov z konca abecedy a symbolu *.

Slovo je reprezentované ako reťazec znakov z konca abecedy. Inverz prvku reprezentujeme veľkým písmenom.

Databáza termov konjugácie (skrátene termov) a ich indexový súbor sa buduje nasledujúcim algoritmom (pseudokód):

```
terms = ATOMIC_TERMS;
len = length(ATOMIC_TERMS);
last_generation_length = length(ATOMIC_TERMS);

for i := 1 to DEPTH do
    for j := (len - last_generation_length) to len do
        for k := 0 to (j - 1)
            t1 = terms[j];
            t2 = terms[k];

            if ((not ld_instance(t1, t2) &&
                (not larue1_instance(t1, t2) &&
```

```
                (not larue2_instance(t1, t2) &&
                (terms_fst_in_list(t1, t2))
            )
                push(terms, ("t1 * t2", "t2 * t1"));

                w1 = translate_term_to_word[t1];
                w2 = translate_term_to_word[t2];
                if (atoms_in_term_lex_ordered(t1))
                    push(words[w1], t1);
                fi
                if (atoms_in_term_lex_ordered(t2))
                    push(words[w2], t2);
                fi
            fi
        done
    done
done
```

Začíname teda od atomických termov. V každom kroku vezmeme naposledy vygenerovanú generáciu termov a skombinujeme ju s generáciami vygenerovanými pred ňou. Avšak kombináciu termov {`t1, t2`} vynechávame v týchto prípadoch:

`ld_instance(t1, t2)`
: výsledný term je v tvare pravej strany ľavej distributivity (ak máme vygenerovať pravú stranu, museli sme vygenerovať ľavú strany $LD$, preto by bol tento term redundantný)

`larue1_instance(t1, t2), larue2_instance(t1, t2)`
: z rovnakého dôvodu vynechávame pravú strany prvých dvoch Larueových rovníc

`terms_fst_in_list(t1, t2)`
: obidva termy musia byť prvé zo zoznamu na svojom indexe, t.j. `t1` ani `t2` nie sú pravou stranou nejakej rovnice, ktorá platí v grupe [1]

Nový term indexujeme iba v prípade, že prvé výskyty jeho atómov sú lexikograficky usporiadané, tak sa vyhýbame rovniciam, ktoré sa líšia iba pomenovaním premenných.

[1] Posledné tri testy boli implementované až pred vykonaním výpočtov popísaných v kapitole 4.2

## 3.2 Generovanie rovníc

Z databázy slov môžeme vyextrahovať zoznam rovníc konjugácie splnených v grupe.

Do zoznamu rovníc nepridávame rovnice, ktoré plynú tranzitivitou z už pridaných rovníc a rovnice tvaru $t1 * t2 = t1 * t3$ a $t2 * t1 = t3 * t1$, ktoré sú ekvivalentné rovnici $t2 = t3$ (pretože grupoid $G(*)$ je zľava i zprava divizibilný, viď [1]).

Pseudokód:

```
equations = array();

foreach w from words do
    terms = words[w];
    for i:= 1 to count(terms) - 1 do
        for j:= 1 to count(terms) do
            t1 = terms[i];
            t2 = terms[j];
            (l1, r1) = decompose_to_operands(t1);
            (l2, r2) = decompose_to_operands(t2);

            if (l1 != l2) and (r1 != r2)
                push(equations, "terms[i] = terms[j]");
            fi
    done
done
```

V programe je rovnica reprezentovaná reťazcom `"term1=term2"`, kde `term1` a `term2` sú v infixe.

## 3.3 Dokazovanie rovníc

Súbor rovníc vygenerovaný skriptom `eq_generator.pl` (viď sekcie 2.2 a 3.2) môže byť vstupným súborom skriptu `eq_prover.pl`.

V každom kroku vytvoríme vstupné súbory v syntaxi externých dokazovačov a nové procesy, ktoré spustia externé dokazovače (takže dokazovače bežia paralelne). Potom, kým bežia nejaké detské procesy, opakujeme:

1. Počkáme na dokazovač, ktorý ukončí svoju úlohu v časovom limite ako prvý.

2. Zaznamenáme návratový kód dokazovača.

3. Ak hľadanie skončilo úspechom, pošleme ostatným bežiacim procesom dokazovačov signál `SIGINT`.

   Ak dokazovač skončil neúspechom (typicky timeout), prejdeme ku kroku 1 (teda čakáme na výsledkých ďalších bežiacich dokazovačov).

   V závislosti na prepínači `-successful-results` a návratovom kóde dokazovača odstránime alebo ponecháme vstupné a výstupné súbory.

Abstraktná trieda `ProverBase` poskytuje metódy na vytvorenie vstupného súboru pre dokazovač, spustenie dokazovača, odstránenie vstupných a výstupných súborov v prípade úspechu, odstránenie nadbytočných bežiacich procesov, zaznamenanie návratového kódu procesu a vypísanie štatistiky všetkých ukončených procesov dokazovača.

Triedy `ProverP9`, `ProverWM`, `ProverM4` implementujú tieto metódy pre konkrétne dokazovače. V hlavnom cykle tak v každom kroku inštanciujeme triedy na manipuláciu dokazovačov, ktoré chceme použiť a využívame polymorfizmu objektov. Takto môžeme doimplementovať použitie nového externého dokazovača bez zásahov do kódu hlavného cyklu spracovania rovníc.

Metódy, ktoré spustia externý dokazovač, môžeme v prípade skoršieho úspechu iného dokazovača ukončiť, avšak proces dokazovača ostane v systéme bežať na pozadí. Preto má každá trieda na manipuláciu externého dokazovača metódu `killall` na odstránenie týchto nežiadúcich procesov.

Východzie hodnoty pre časové limity, adresáre pre vstupné a výstupné súbory a pod. sú definované v module `src::Config`.

Pri dokazovaní používame (okrem časového limitu) východzie voľby dokazovačov.

# Kapitola 4

# Prehľad výsledkov

Skúmali sme rovnice o troch premenných vygenerované z množiny termov hĺbky 5 a rovnice o štyroch premenných vygenerované z 5% termov hĺbky 5.

## 4.1 Rovnice o troch premenných

Zoznam všetkých vygenerovaných rovníc o troch premenných je v prílohe v súbore `3var/equations.txt`. Databáza a index binárnych termov sú v súboroch `3var/terms.txt` a `3var/words.db`.

Medzi nimi dokázateľne neexistuje rovnica, ktorá by spolu s rovnicami ľavej distributivity a idempotencie implikovala aspoň prvé dve z Laruových rovníc (to znamená, že program *Mace4* našiel model pre každú rovnicu testovanú na cieľ 1). Našli sme však 218 rovníc, ktoré nie sú dôsledkom $Larue_1$ a $Larue_2$. Tieto rovnice sú v súbore `3var/ld+i+larue1,2-model.txt`.

Nasledujúce dve patria medzi najkratšie z nich:

$$\begin{aligned} R_1 &\equiv ((xy * x) * (y * xy)) * ((xy * x) * z) = ((x * yx) * y) * xz \\ R_2 &\equiv ((x * xy) * y) * (x * (x * xy)) = (x * xy) * (yx * (yx * xy)) \end{aligned}$$

Uvažujme konjunkciu

$$Larue_1 \,\&\, Larue_2 \,\&\, \ldots \,\&\, Larue_{10} \,\&\, \neg R_1$$

a ľavodistributívny idempotentný grupoid $G(*)$ s nasledujúcou interpretáciou *:

| * | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 2 | 3 | 4 | 2 | 0 | 3 |
| 1 | 3 | 1 | 5 | 6 | 1 | 3 | 5 |
| 2 | 4 | 3 | 2 | 0 | 3 | 4 | 2 |
| 3 | 2 | 5 | 4 | 3 | 0 | 6 | 1 |
| 4 | 3 | 4 | 0 | 2 | 4 | 3 | 0 |
| 5 | 5 | 6 | 3 | 1 | 6 | 5 | 3 |
| 6 | 1 | 3 | 6 | 5 | 3 | 1 | 6 |

Ak položíme $x = 0$ a $y = 1$, konjunkcia je splnená. Záznam strojového hľadania z programu *Mace4* je v prílohe pod názvom `3var/r1.out`.

Podobne konjunkcia

$$Larue_1 \,\&\, Larue_2 \,\&\, Larue_3 \,\&\, \neg R_2$$

je splnená v nasledujúcom ľavodistributívnom idempotentnom grupoide:

| * | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| 0 | 0 | 2 | 1 | 4 | 3 | 5 |
| 1 | 3 | 1 | 4 | 5 | 2 | 3 |
| 2 | 4 | 3 | 2 | 1 | 5 | 4 |
| 3 | 2 | 4 | 5 | 3 | 1 | 2 |
| 4 | 1 | 5 | 3 | 2 | 4 | 1 |
| 5 | 0 | 2 | 1 | 4 | 3 | 5 |

keď položíme $x = 0$, $y = 1$ a $z = 0$. Záznam programu *Mace4* je v súbore `3var/r2.out`.

Vyvstáva otázka, či takýto model existuje v prípade $R_1$ alebo $R_2$ pre všetky $Larue_n, n \geq 1$.

Našli sme iba jednu rovnicu spĺňajúcu cieľ 3:

$$R_3 \equiv ((x * yx) * (yx * x)) * (xy * z) = ((x * yx) * yx) * ((x * xy) * z)$$

pre ktorú platí:

$$LD \,\&\, I \,\&\, Larue_1 \,\&\, R_3 \Rightarrow Larue_2$$

Dôkaz dokazovača *Prover9* je v súbore `3var/r3.out`.

Žiadna zo vstupných rovníc nespĺňa cieľ 4.

## 4.2 Rovnice o štyroch premenných

Z dôvodov výpočtovej náročnosti sme proces generovania termov zastavili po vygenerovaní 5% z celkového počtu termov. Generovanie trvalo približne tri dni na procesore AMD Athlon$^{TM}$64 3000+. Databázy termov a slov sú v súboroch `4var/terms.txt` (1.9 GB) a `4var/words.db` (466 MB). Zoznam testovaných rovníc je v súbore `4var/equations.txt`.

Nenašli sme ani jednu rovnicu $R$, ktorá by spĺňala podmienky 2–4, pričom sa nepodarilo vyvrátiť, že rovnica spĺňajúca ciele 3 a 4 neexistuje. Rovnice, pri testovaní ktorých sa nepodarilo nájsť príslušný model v limite 20 sekúnd sú v súboroch `4var/ld+i+e+larue1-implies-larue2-nomodel.txt` a `4var/ld+i+e+larue2-implies-larue1-nomodel.txt`.

Predkladáme však rovnicu

$$(xy * x) * ((yx * y) * zu) = ((xy * x) * y) * ((xz * x) * (xz * u))$$

ktorá nevyplýva z množiny premís $\{LD, I, Larue_1, Larue_2\}$. Dôkaz sa nachádza v textovom súbore `4var/r1.out`. Domnievame sa, že táto rovnica nevyplýva dokonca z $\{LD, I, Larue_1, \ldots, Larue_n\}$ pre všetky $n \geq 1$. Ďalšie nájdené rovnice spĺňajúce túto podmienku boli po substitúcii ekvivalentné rovniciam uvedeným v sekcii 4.1.

## 4.3 Zhrnutie

Prvý z hlavných cieľov deklarovaných v kapitole 1, teda nájsť kandidátov na nové rovnice konjugácie, sa podarilo splniť – našli sme tri rovnice, ktoré neplynú z množiny $\{LD, I, Larue_1, Larue_2\}$ a podľa slov vedúceho práce je pravdepodobné, že bude nájdený ručný dôkaz, že neplynú z $\{LD, I, Larue_1, Larue_2, \ldots\}$.

Druhý z cieľov, teda zjednodušiť Larueove rovncie, sa splniť nepodarilo – žiadne rovnice, ktoré by spolu s rovnicami $\{LD, I\}$ implikovali aspoň prvé dve z Larueových rovníc, sme nenašli.

Čo sa týka skúseností s efektivitou použitých systémov dokazovania rovníc, dokazovač *Prover9* bol úspešnejší vo väčšom množstve prípadov, než dokazovač *Waldmeister*.

# Kapitola 5

# Elektronická príloha

Priložené DVD v adresári `Conjug/` obsahuje:

`3var` výsledky výpočtov zo sekcie 4.1

`4var` výsledky výpočtov zo sekcie 4.2

`conf/Config.pm` konfiguračný súbor

`data` adresár s axiómami a cieľovými formulami použitými pre hľadanie dôkazov a modelov

`doc/Systémy automatického dokazování.pdf` text tejto práce

`src` zdrojové kódy v jazyku *Perl*

`eq_generator.pl, eq_prover.pl` spustiteľné skripty pre generovanie a testovanie rovníc v jazyku *Perl*

`eq_postfix_to_infix.pl` skript pre prevod rovníc z postfixu do infixu

`eq_infix_to_postfix.pl` skript pre prevod z infixu do postfixu

`gen_larue.pl` skript pre generovanie larueových rovníc

`list_substract.pl` skript pre množinový rozdiel zoznamov rovníc

# Literatúra

[1] Drápal A., Kepka T., Musílek M. (1994): Group conjugation has non-trivial LD-identities. Comment. Math. Univ. Carolinae, 35/2, 219-222

[2] Larue D. (1999): Left-distributive idempotent algebras. Commun. Alg. 27/5, 2003-2009

[3] Stanovský D. (2003): On Equational Theory of Group Conjugation. Contr. Gen. Alg. 15